Worksheet 4 Subroutines

Task 1

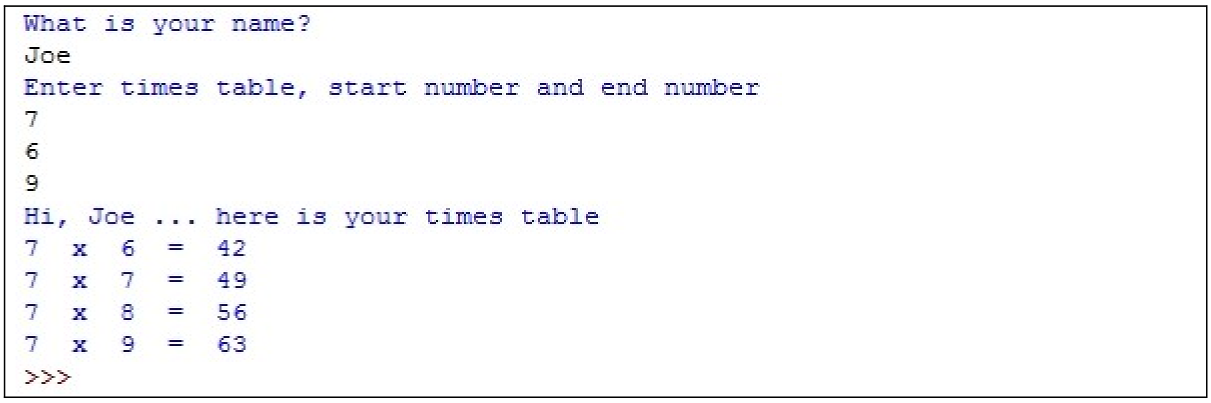
1. A primary school teacher requires a program that will allow pupils to practise their multiplication tables (times tables). The program must allow them to choose the table they want displayed and the start and end numbers to multiply by.

   For example, if the pupil enters 5, 4, 12 the program will display

   5 x 4 = 20
   5 x 5 = 25
   …
   …
   5 x 12 = 60

   The program will then print a message followed by the table selected.

   Create a pseudocode solution for this using a subroutine called **multiples()** which takes **table**, **startnum**, **endnum** and **pupilName** as parameters. The subroutine will output the message and multiplication table. The main program will prompt the user to enter the values and will then pass them to the routine.



```
What is your name?
Joe
Enter times table, start number and end number
7
6
9
Hi, Joe ... here is your times table
7  x  6  =  42
7  x  7  =  49
7  x  8  =  56
7  x  9  =  63
>>>
```

Task 2

2. It is possible to return more than one value from a function. Consider the following Python program:

```
# pass multiple results back from a function
def calc(a,b):
    constantVal = 2
    x = constantVal *(a + b)
    y = a - b
    z = a * b
    return x, y, z

#main program
add, subtract, multiply = calc(5,3)
print (add, subtract, multiply)
```

   (a) Name a **local variable** in the subroutine

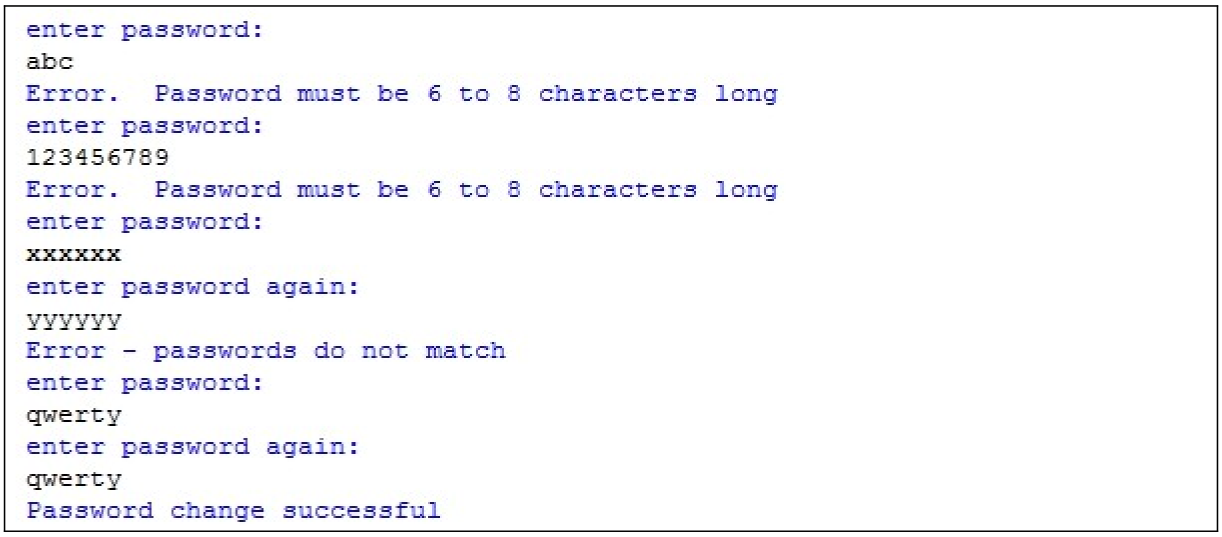
   (b) What is printed by the statement in the final line of the program?

3. It is often necessary to verify information upon data input. One method of doing this is double entry.

   Write pseudocode for a subroutine called **getPword()** that takes one parameter, called **attempt**, which can have a value of 1 or 2. The subroutine should prompt the user to enter a password if attempt = 1, or prompt the user to re-enter a password if the attempt = 2, and then return the password.

   The subroutine should also check that the length of the password is a valid length between 6 to 8 characters. The main program will verify that the two passwords are the same, and re-prompt for entry if they are not. If both passwords are the same, a message is displayed, informing the user that the password change has been successful.

   The output below demonstrates how the program will work.



```
enter password:
abc
Error.  Password must be 6 to 8 characters long
enter password:
123456789
Error.  Password must be 6 to 8 characters long
enter password:
xxxxxx
enter password again:
yyyyyy
Error - passwords do not match
enter password:
qwerty
enter password again:
qwerty
Password change successful
```

4. A company runs a private car park near an airport. The car park has 10 rows numbered 1-20 and each row has spaces (referred to as columns) numbered 1-6 for 6 cars. Customers leave their cars with keys at the car park office, and a driver parks it in a free space.

   The space is referenced by its grid coordinates row and column. E.g. a car parked in the 3rd row, 5th space would have the grid reference [3,5].

   The driver enters the car registration into the computer. A car with registration AVH 61 HU parked at grid reference [3,5] would assign "AVH 61 HU" to **park[3,5]**. Empty spaces are denoted, for example, by **park[3,5] = "empty"**

   Write pseudocode for a program which displays a menu with 5 options, and for the first four options, calls the relevant subroutine.

   Option 1: Set all spaces in the car park to "empty"

   Option 2: Park a car. This option asks the user to enter the registration number of a car and the grid reference (row and column number) where it has been parked. The program checks that this is an empty space, and if it is, puts the registration number in the appropriate element of the array. If it is not, it asks the user to enter the grid reference.

   Option 3: Remove a car. This option asks the user to enter the registration number, searches the grid for the number and then resets it to "empty".

   Option 4: Display the car park grid

   Option 5: Quit

   Pseudocode for the main program is given below. This initialises the car park grid to "empty" and then repeatedly displays the menu of options and performs the required function until the user selects "Quit".

   Write subroutines for options 2-5. (Assume array indices start at 0)

```
// main program
initialise car park grid to "empty"
#display menu of options
print("1. Reset all spaces in the car park to 'empty'")
print("2. Park a car")
print("3. Remove a car")
print("4. Display the car park grid")
print("5. Quit\n")
option = input("Enter your choice: ")
// accept choice
while option != "5"
    if option == "1"
        emptyCarPark(carPark)
    else if option == "2"
        parkACar(carPark)
    else if option == "3"
        removeACar(carPark)
    else if option == "4"
        displayCarParkGrid(carPark)
    else
        option = ("Invalid choice - please re-enter: ")
   endif
    print("1. Reset all spaces in the car park to ‘empty’")
    print("2. Park a car")
    print("3. Remove a car")
    print("4. Display the car park grid")
    print("5. Quit\n")
    option = input("Enter your choice: ")
endwhile
print("Goodbye")
```